

An Example of Relational Schema Normalization

Sudarshan S. Chawathe

Computer Science Department, University of Maryland, College Park, MD 20742, USA.
chaw@cs.umd.edu

October 16, 2003

This note outlines the normalization process applied to a typical relational schema. There is nothing special about this example other than it being the one I discussed in class and it serving quite well to illustrate the normalization process.

We begin with a fairly pedestrian relational schema for the familiar domain of students and courses:

Students(SID, name, phone, cdept, cnum, cname)

The attributes cdept, cnum, and cname refer to the department (e.g., CMSC), number (e.g., 424), and name (e.g., Database Design) of a course, while the SID attribute uniquely identifies a student. We assume that a student has exactly one name but may have any number of phone numbers listed in our database. Phones may be shared. We permit courses in different departments to have the same course numbers or the same course names. However, we assume that within a department, either a course number or a course name suffices to uniquely identify a course. Given these assumptions about our domain, we may determine the following functional dependencies:

$f_1: \text{SID} \rightarrow \text{name}$
 $f_2: \text{cdept, cnum} \rightarrow \text{cname}$
 $f_3: \text{cdept, cname} \rightarrow \text{cnum}$

Some plausible dependencies that do *not* hold in our domain are indicated below. Of course, there are many more non-dependencies.

$\text{name} \nrightarrow \text{SID}$
 $\text{SID} \nrightarrow \text{phone}$
 $\text{phone} \nrightarrow \text{SID}$
 $\text{cnum} \nrightarrow \text{cname}$
 $\text{cnum} \nrightarrow \text{cdept}$

Using the dependencies f_1 , f_2 , and f_3 , we may infer that the keys of the Students relation are {SID, phone, cdept, cnum} and {SID, phone, cdept, cname}. (Verify that there are no other keys.)

The dependency f_1 is a BCNF violation for Students because it is nontrivial and its left-hand side (SID) is not a superkey. (So are f_2 and f_3 .) We decompose using f_1 to obtain the following schema:

Students1(SID, name)
Students2(SID, phone, cdept, cnum, cname)

Students1 is in BCNF because it is binary. (Verify that all binary relations are in BCNF. Hint: Perform a case analysis on the functional dependencies that may exist for such a relation.)

Students2 has the functional dependencies f_2 and f_3 . (Check that these dependencies hold and that there are no others.) Thus, the keys for this relation are {SID, phone, cdept, cnum} and {SID, phone, cdept, cname}. Both f_1 and f_2 are BCNF violations. We decompose using f_1 to yield the following schema as the replacement for Students2 (but not Students1):

Students21(cdept, cnum, cname)
Students22(cdept, cnum, SID, phone)

The functional dependencies for Students21 are f_2 and f_3 . The keys are {cdept, cnum} and {cdept, cname}. Since the left-hand sides of both dependencies are superkeys, Students21 is in BCNF. There are no nontrivial functional dependencies in Students22. Therefore, it is trivially in BCNF.

We have thus reached the end of the BCNF normalization process. The final schema consists of the relations Students1, Students21, and Students22. By choosing more appropriate names and reordering some of the attributes, we arrive at the following:

Students(SID, name)
Courses(cdept, cnum, cname)
Enrollments(SID, phone, cdept, cnum)

One may observe that, while the names Students and Courses seem quite fitting for the corresponding relations, there is something strange about using the name Enrollments for the third relation. One would expect a relation recording enrollment information to list students and the classes they take, perhaps along with supplementary information relevant to the enrollment (such as the number of units of credit). It seems strange that the phone number, which is not directly related to a course or an enrollment, is included in the Enrollments relation. We may suspect that we have made an error. However, checking our work and repeating the exercise from first principles reveals

that the relation is indeed in BCNF because it has no functional dependencies. This observation suggests that functional dependencies and BCNF are not sufficient to model desirable schemas, and we must use multivalued dependencies and 4NF.

The above discussion suggests the multivalued dependency $SID \twoheadrightarrow phone$. Informally, this dependency states that each SID is associated with a unique set of phone numbers that is independent of the other attributes in the relation (cdept and cnum). This statement seems valid given our domain. We may also confirm the validity of this dependency using the formal definition: If we know that tuples (i_1, p_1, d_1, n_1) and (i_1, p_2, d_2, n_2) belong to the Enrollments relation, can we conclude that (i_1, p_1, d_2, n_2) also belongs to the relation? (Check by using a sample instance as a thinking aid. Of course, dependencies do not follow from the instances and we must use instances only as guides to confirm some statement about our domain.)

Since Enrollments has no nontrivial functional dependencies, its only key is the set of all its attributes. (Note that although we are considering multivalued dependencies, keys are still defined only in terms of functional dependencies.) It follows that $SID \twoheadrightarrow phone$ is a fourth normal form violation. Decomposing using this dependency yields the following to replace the Enrollments relation:

Enrollments1(SID, phone)
Enrollments2(SID, cdept, cnum)

Enrollments1 is in 4NF because it is binary. (Verify that all binary relations are in 4NF). Enrollments2 has no nontrivial dependencies (functional or multivalued) and therefore is also in 4NF. We have thus reached the end of the process of normalization to 4NF. The final schema, using more intuitive names for Enrollments1 and Enrollments2 is as follows:

Students(SID, name)
Courses(cdept, cnum, cname)
Phones(SID, phone)
Enrollments(SID, cdept, cnum)

Exercises

1. Verify the claims made above regarding the functional dependencies and keys.
2. Often, we have a choice of which functional dependency to use for decomposing a relational schema that is not in BCNF. Repeat the normalization procedure by choosing different dependencies whenever possible.

3. Use a completely mechanical process to normalize the original relational schema. That is, everything after the *initial* determination of functional dependencies for the Students relation should be performed in a manner that could be replicated by a simple program.
4. Design an ER model that, when mapped to a relational schema using the standard methods described in the textbook, directly yields the final 4NF schema above. (This exercise in reverse engineering is only for improving our understanding, and not the recommended method for designing ER models!)
5. Design an ER model that, when mapped as in Exercise 4, yields a relational schema that is different from the final scheme in our example. Transform that schema, if necessary, to 4NF.